

Fahrenheit Co.

Communities

Project Demonstration Report

Eric Anderson - Implementation, Functionalities & References
Christopher Bednarz - Level 3 Diagram and UML
Zoe Chamlee - Performance Metrics
James W. Garrett - Editing
Mason Holley - Code Blocks
Rishabh Tewari - Gantt Chart & Future Plans

Date of Submission: November 8th, 2021

Executive Summary

Communities is an app which allows people within a local scene to have an easier time interacting with each other on a more personal level. Our program does this by isolating the scope of what the client can see to their local areas. This app consists of a map of the user's local area that will allow them to browse and join local groups and communities. They will be able to navigate the groups they join via a feed that will post all info related to the communities.

Introduction & Brief Explanation

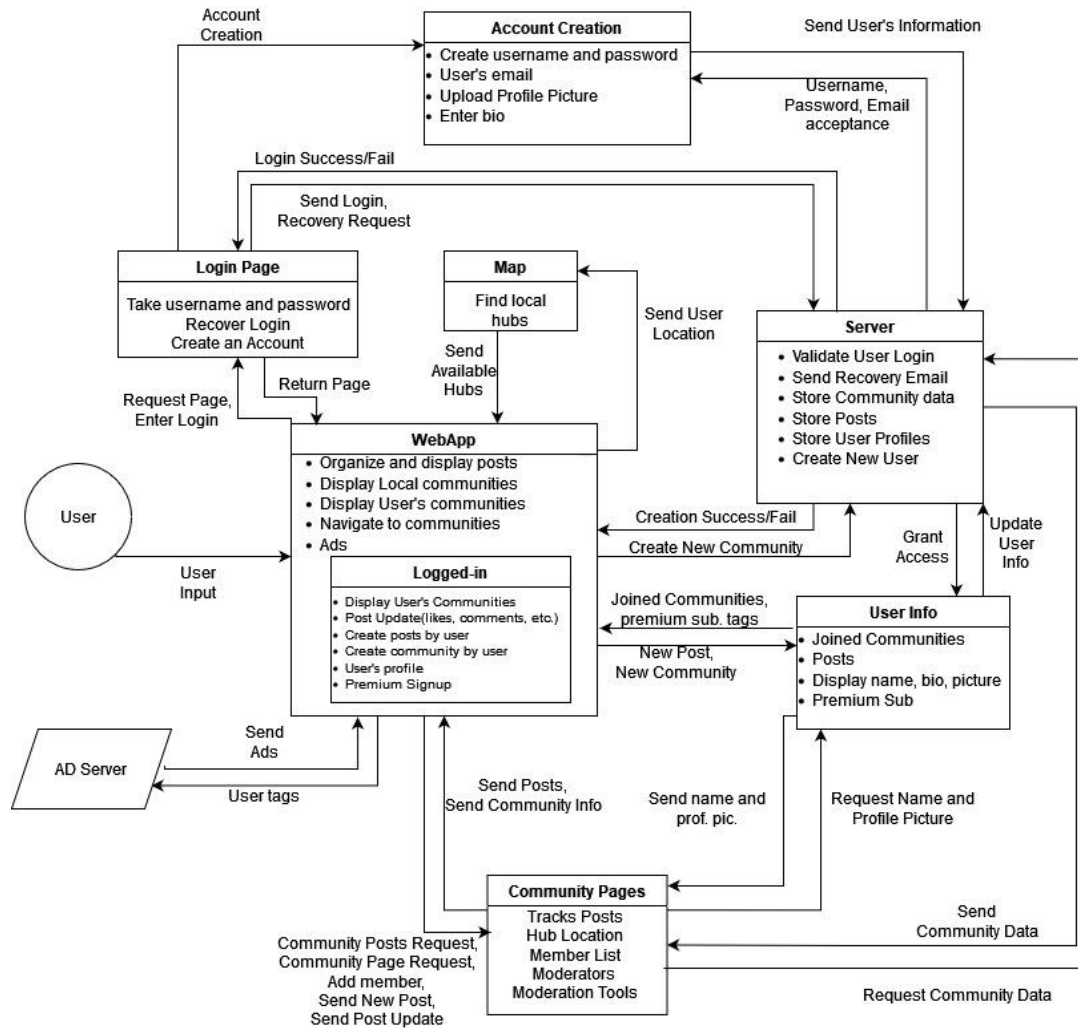
Our focus in creating Communities is to provide a platform for local communities to grow and flourish in a healthy environment. Our app, Communities, will allow users to connect with or create a local scene for all kinds of interests, hobbies, or activities. With each user that joins our platform, we gain a new potential member for one of the many communities. While a user might join for one specific community, we want them to be able to find all the communities which fit their tastes in their local area. With this, we believe our company, Fahrenheit Co., will be able to cultivate a healthy user-base which will allow continuous app growth.

We hope to diversify from other similar applications such as Discord and Reddit by location-locking our communities. Our main goal is to prevent our app from having obscenely large and detached communities. Each community will have the tools available to announce and schedule events related to each community which will be easily accessible and known to all members. We will also be working to provide tools to allow the creators or appointed leaders of a community to maintain and moderate the community.

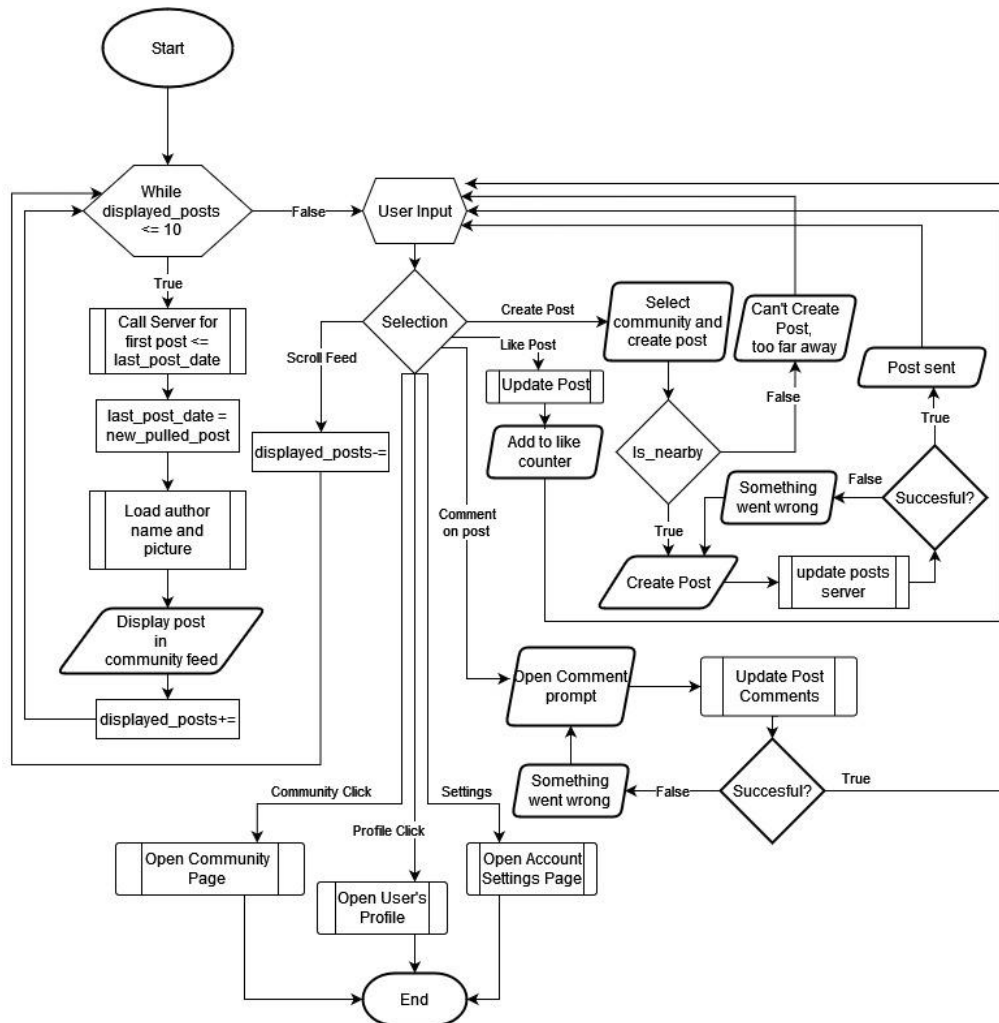
Level 3 Diagram and System Overview

As shown by the level 3 diagram below, each individual process is designed to work with each other. With this design, we can prevent one part from breaking the others in the event of an error. This strategy is helpful for full-scale deployment scenarios and narrowing down bug locations. As you can see, the core functionality of our app allows for user accounts, local communities, and community posts to be created and displayed to the appropriate people.

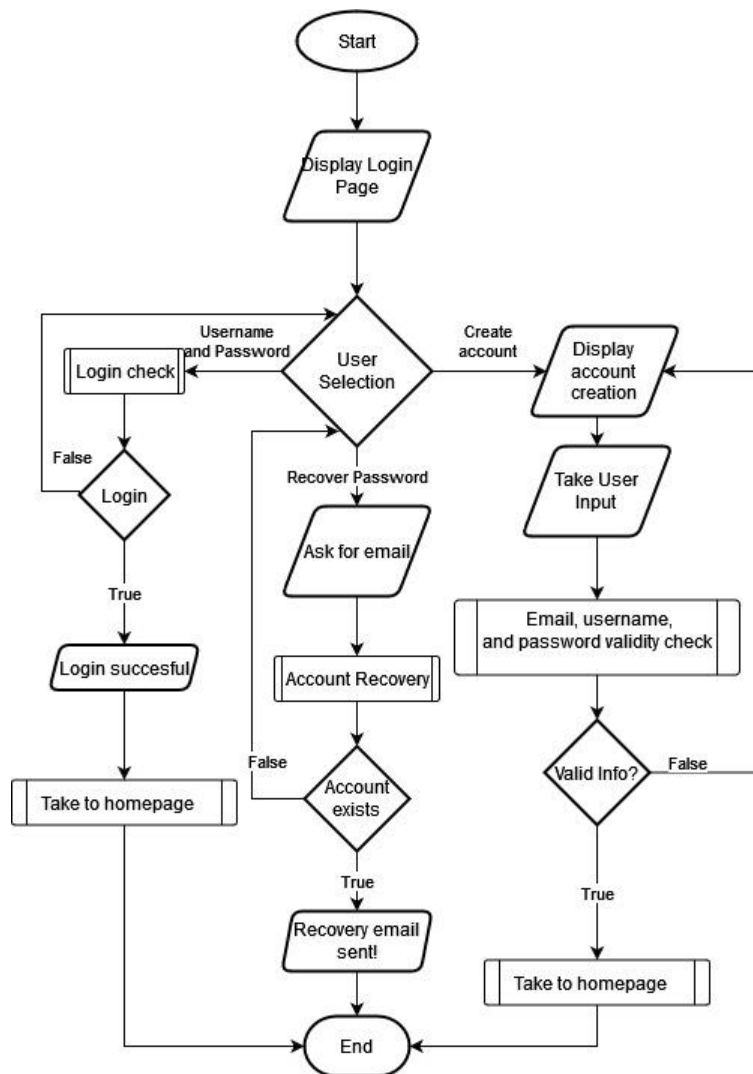
Each of these features piggybacks off one another, so for instance if a user must have an account to create a post, and any given post needs to have an attached community with it, and the users must be within said community's surrounding area in order to post the aforementioned posts.



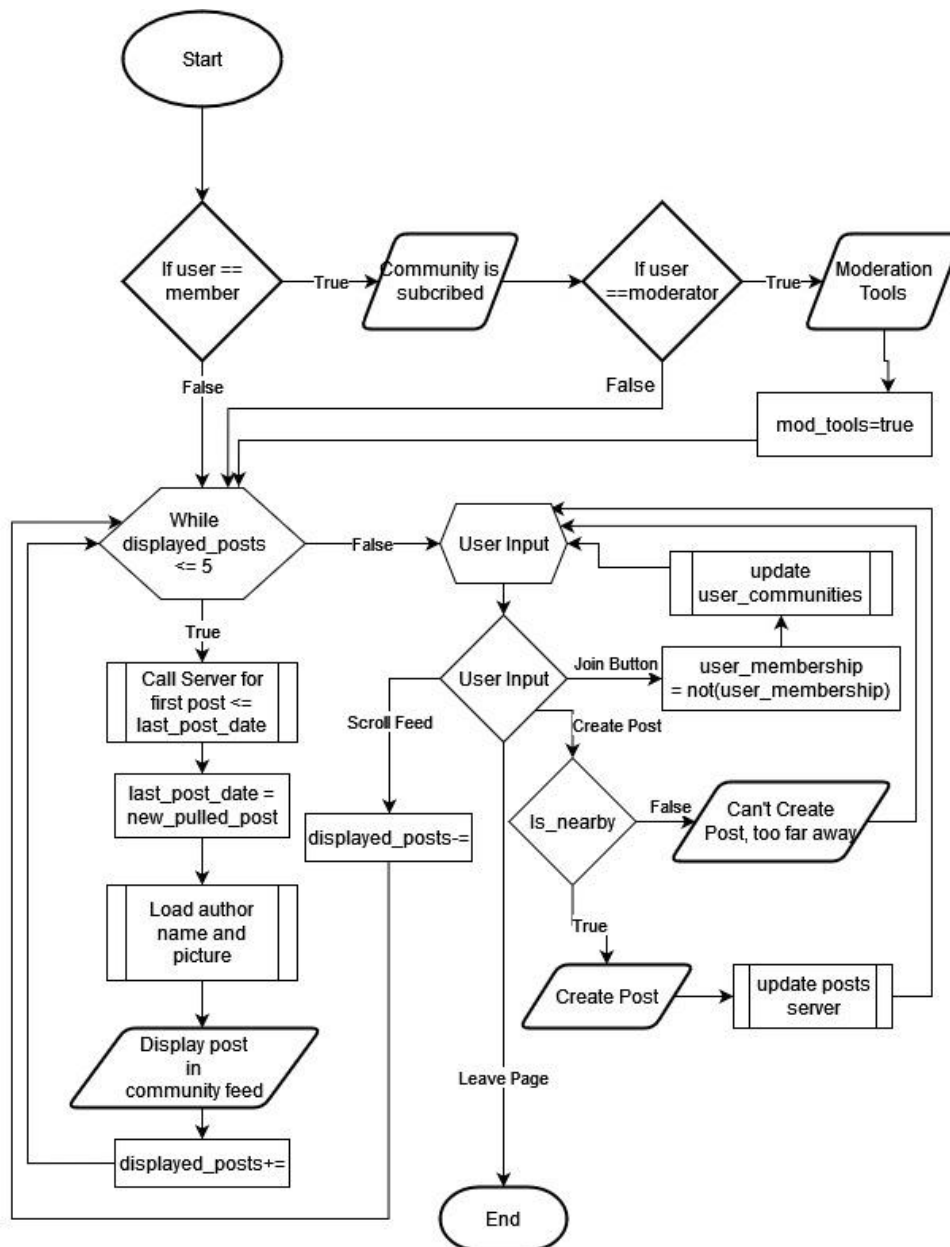
UML to further clarify the system



This flowchart shows the home page's sequence of events. The page will display posts, dependent on the user's subscribed communities and location before giving the user their choice of actions to take, be that creating a new post, visiting a community or loading a new set of posts.



This flowchart shows the logging in process. The process should allow the user to create an account, check for unique email and username, test to see if a password is valid, login, or recover their password. Once any of these actions are complete, the corresponding page will load (either the homepage or login page once again.)



This flowchart demonstrates the functionality of the Community pages. Similar to the homepage, this will display posts specific to that community, from which the user can join the community, create a post, or load more posts, or if the user is a moderator they can also use their special tools to moderator the community.

System Implementation and Testing

The system was implemented via Revel, our chosen web framework. Revel is a full stack framework that utilizes Go for the backend, and HTML for the frontend. We also utilized Mariadb for the creation of our database of users, communities, and posts.

Whenever we wanted to run a test on our application, we first spin up the server that is holding our database and then run our application via the terminal command `revel run Communities`. The command builds our website and allows us to test any features that we wanted to run.

Significant Code Snippets and Explanation

```
//By default, Index is the first page that loads in Revel
//We are using this to open up our database and make queries.
func (c App) Index() revel.Result {
    //Error that will display if the database connection fails
    var err error

    //Opening the connection to the database
    //This assumes that the database username and password is root
    db, err = sql.Open("mysql", "root:root@tcp(127.0.0.1:3306)/serverstorage")

    //If database fails to connect, display the error mentioning that the database failed to connect
    if err != nil {
        panic(err.Error())
        c.Flash.Error("Database failed to load")
    }

    //Load the communities nearby
    LoadAllCommunities()
    LoadAllPosts()

    //Ping the database in order to ensure that it is connected
    db.Ping()

    //After connecting to the database, redirect to the Login page
    return c.Redirect(App.Login)
}
```

This block covers our Revel implementation which allows us to connect to the database and display the relevant data to our front-end. It first opens our database and, if successful, it grabs the relevant communities and posts and then displays them on our front-end. After it connects to the database, it redirects the user to the login page.

```
//Handles Account Creation
//Called whenever a "CreateAccount" form is submitted
func (c App) CreateAccount(NewUserName string, NewPassword string, NewEmail string, NewPasswordConfirmation string) revel.Result{
    //If passwords do not match, redirect to the Account Creation page
    if(NewPassword != NewPasswordConfirmation){
        c.Flash.Error("Passwords do not match.")
        return c.Redirect(App.AccountCreation)
    }else if(DBCreateAccount(NewUserName, NewPassword, NewEmail, CurrentSess)){
        // If the creation of the account is successful, redirect to the login page.
        c.Flash.Success("Account Created! You may login now. ")
        return c.Redirect(App.Login)
    }
    //If an error occurred when creating the account, return to the account creation page.
    c.Flash.Error("Error occurred when creating the account, email or username already exists.")
    // defer db.Close()
    return c.Redirect(App.AccountCreation)
}
```

This block handles account creation. It validates the entered password and returns a success message if it successfully communicates with our database and stores the new login. It then checks to see if the username is already in use and redirects back to the account creation with a success message if successful.

```
//Creates the map for the home page
func createMap(lat float64, lng float64) {
    ctx := sm.NewContext() //Creates a new map 'context'
    ctx.SetSize(1080, 1080) //Sets the size of the map in pixels(?)

    //This should change depending on the city, for now, default to 11.
    ctx.SetZoom(11)

    ctx.AddObject(
        sm.NewCircle( //Draw the circle on the map
            s2.LatLngFromDegrees(lat, lng), //Position by latitude / longitude
            color.RGBA{17, 104, 151, 0xff}, //Outline color
            color.RGBA{150, 212, 231, 1.0}, //Fill color
            20000.0, //Radius
            1, //Weight
        ),
    )

    ctx.AddObject(
        sm.NewMarker( //Draw a marker on the map
            s2.LatLngFromDegrees(lat, lng), //Position by latitude / longitude
            color.RGBA{0xff, 0, 0, 0xff}, // Color of marker
            16.0, //Size of the marker
        ),
    )

    img, err := ctx.Render() //Converts the map data to an image on OSM
    if err != nil {
        panic(err)
    }

    if err := gg.SavePNG("public/img/my-map.png", img); err != nil { //Downloads the image from Open Street Map
        panic(err)
    }
}
```

This block handles the map implementation. Currently, it creates an Open Street Map context at a given latitude and longitude. It then sets the size, zoom, and relevant markers. For the markers, it adds a circle to the map that shows the radius of the city the user is currently in. It also adds a marker in the center of that city to allow the user to

differentiate between the different cities on screen. Finally, it renders the map into a temporary image file for ease of use in our front end.

Performance Measuring

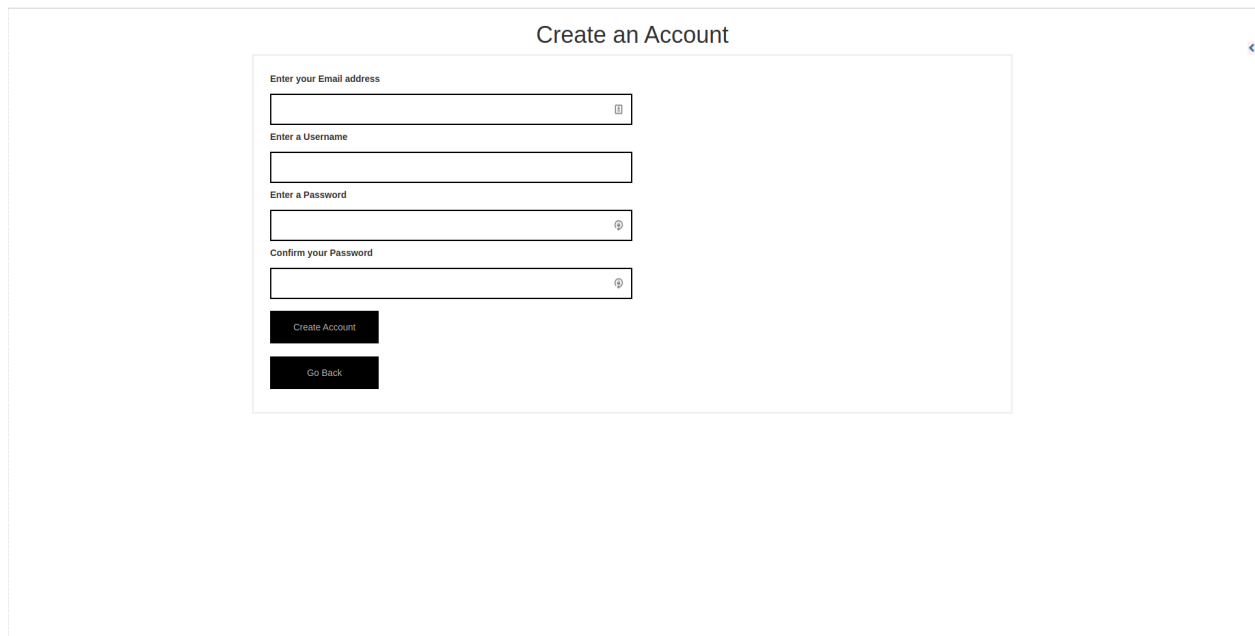
Performance was measured primarily via a small helper function that utilizes the defer functionality of go and standard go logging. This was used since, in the go Revel backend framework, all functionalities are very compartmentalized, for instance logging in on our website is done almost entirely within the scope of a function called `DBLogin()`. By measuring the time for it to return, we can determine how long it takes for the backend to perform a login. This is specifically accomplished by calling `defer finishPerfMeasure(start time.Time, name string)` which in turn calls `finishPerfMeasure()` at the END of the function scope, a helper function.

`startPerfMeasure()` returns the current timestamp which is called immediately so that when the `finishPerfMeasure()` call is performed. At this point, we use the start time and compare to the finishing time and then log this in the performance log file under the function name specified by the name argument. Our average times come in at 1.52ms for logging a user in, 1.38ms for loading their communities, and a very low 0.31ms for loading all of their posts although as image media is added for posts and our database grows this is expected to grow appropriately.

```
[PERFORMANCE] 2021/11/01 19:33:24 DBLogin() execution time: 0.002092
[PERFORMANCE] 2021/11/01 19:33:24 LoadAllCommunities() execution time: 0.001065
[PERFORMANCE] 2021/11/01 19:33:24 LoadAllPosts() execution time: 0.000195
[PERFORMANCE] 2021/11/01 20:00:33 LoadAllCommunities() execution time: 0.003424
[PERFORMANCE] 2021/11/01 20:00:33 LoadAllPosts() execution time: 0.000340
[PERFORMANCE] 2021/11/01 20:01:29 LoadAllCommunities() execution time: 0.000829
[PERFORMANCE] 2021/11/01 20:01:29 LoadAllPosts() execution time: 0.000257
[PERFORMANCE] 2021/11/01 20:01:53 DBLogin() execution time: 0.001263
[PERFORMANCE] 2021/11/01 20:01:53 LoadAllCommunities() execution time: 0.000966
[PERFORMANCE] 2021/11/01 20:01:53 LoadAllPosts() execution time: 0.000204
[PERFORMANCE] 2021/11/01 20:02:00 LoadAllCommunities() execution time: 0.000627
[PERFORMANCE] 2021/11/01 20:02:00 LoadAllPosts() execution time: 0.000191
[PERFORMANCE] 2021/11/01 20:02:53 DBLogin() execution time: 0.001231
[PERFORMANCE] 2021/11/01 20:02:53 LoadAllCommunities() execution time: 0.001718
[PERFORMANCE] 2021/11/01 20:02:53 LoadAllPosts() execution time: 0.000789
[PERFORMANCE] 2021/11/01 20:04:16 LoadAllCommunities() execution time: 0.001001
[PERFORMANCE] 2021/11/01 20:04:16 LoadAllPosts() execution time: 0.000203
[PERFORMANCE] 2021/11/01 21:18:15 LoadAllCommunities() execution time: 0.005567
[PERFORMANCE] 2021/11/01 21:18:15 LoadAllPosts() execution time: 0.000364
[PERFORMANCE] 2021/11/02 02:33:17 LoadAllCommunities() execution time: 0.002999
[PERFORMANCE] 2021/11/02 02:33:17 LoadAllPosts() execution time: 0.000376
[PERFORMANCE] 2021/11/03 17:33:37 LoadAllCommunities() execution time: 0.036611
[PERFORMANCE] 2021/11/03 17:33:37 LoadAllPosts() execution time: 0.000921
[PERFORMANCE] 2021/11/03 17:41:02 LoadAllCommunities() execution time: 0.002315
[PERFORMANCE] 2021/11/03 17:41:02 LoadAllPosts() execution time: 0.000343
[PERFORMANCE] 2021/11/03 17:43:57 LoadAllCommunities() execution time: 0.001894
[PERFORMANCE] 2021/11/03 17:43:57 LoadAllPosts() execution time: 0.000224
```

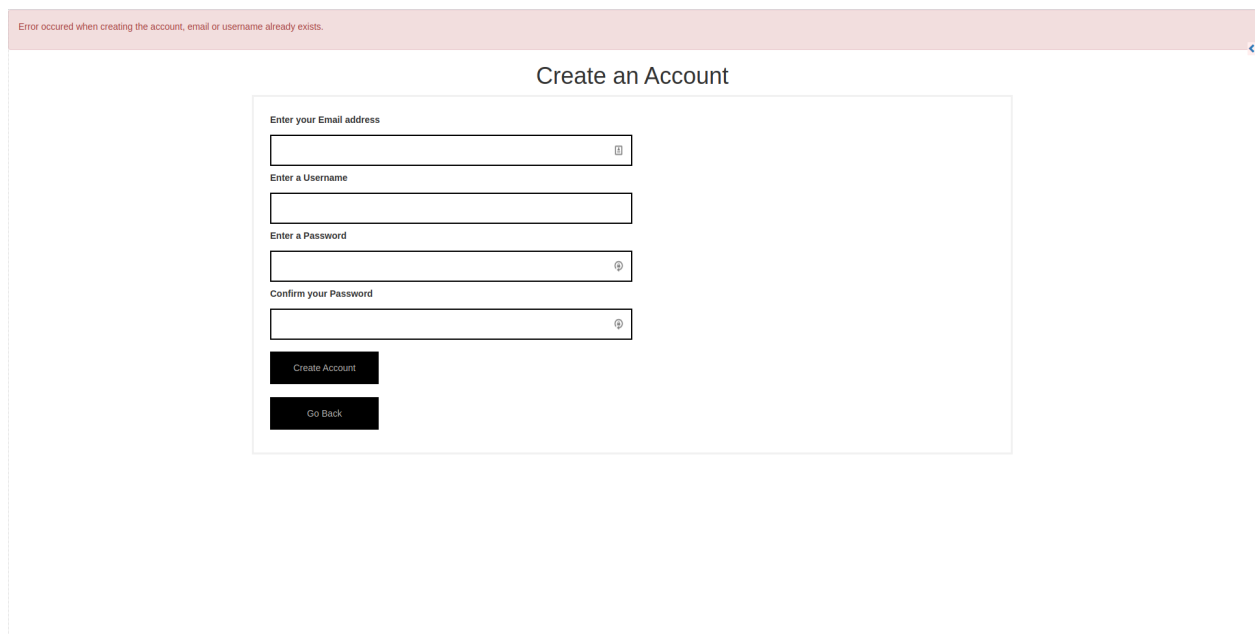
Screenshot of performance log

Evidence of Functionalities of the Working Project



The screenshot shows a web form titled "Create an Account". It contains five input fields: "Enter your Email address", "Enter a Username", "Enter a Password", and "Confirm your Password". The "Enter a Password" and "Confirm your Password" fields have a small eye icon to the right, indicating a password toggle. Below the input fields are two buttons: "Create Account" and "Go Back".

This is our Account Creation page. Users can create an account with an email address, username, and password. The password is prompted twice in order to prevent accidental inputs in it.



The screenshot shows the same "Create an Account" form as above, but with an error message displayed at the top: "Error occurred when creating the account, email or username already exists." The form fields and buttons remain the same.

If a user attempts to create an account with an email or username that is used by another user, the error message "Error occurred when creating the account: Username or email already exists" displays.

Passwords do not match.

Create an Account

Enter your Email address

Enter a Username

Enter a Password

Confirm your Password

Create Account

Go Back

Similarly, if a user does not input the same password on both the “Enter a password” and “Confirm your Password” fields, the error message “Passwords do not match” displays.

Account Created! You may login now.

Welcome to Communities!

Username

Enter Username

Password

Enter Password

Login

☒ Remember me

[Forgot password?](#)

[Create an account.](#)

When a user has successfully created an account, he/she is redirected to the login page with the message “Account Created! You may login now”.

Invalid Username or Password

Welcome to Communities!

Username

Enter Username

Password

Enter Password

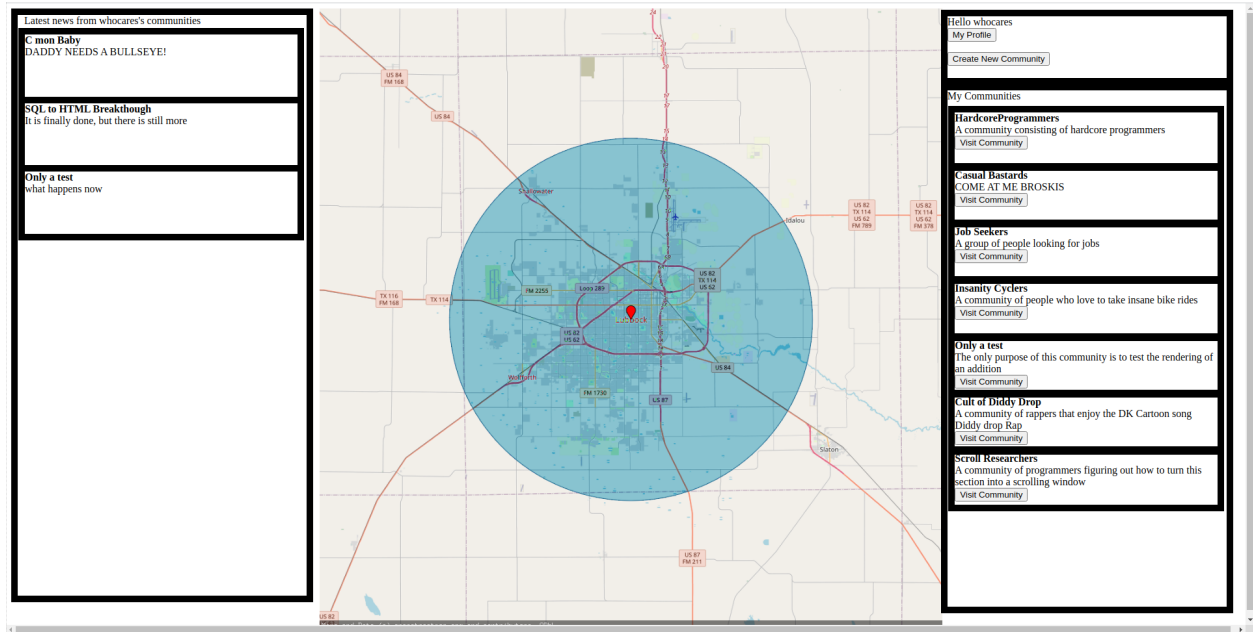
Login

☒ Remember me

[Forgot password?](#)

[Create an account.](#)

When the user inputs a username or password that does not line up with their credentials, the message “Invalid Username or Password” displays.



Once the user logs in, he/she is taken to the home page. The latest posts are displayed on the left, the map is displayed in the center, and the profile window/Communities page displayed on the right. Currently, the map only shows the area of the user. On the final release, the map will show the locations of communities that the user can subscribe to.

My Profile

Current Username:

Change Username
👤

Update Username

Current Password:

Change Password
🔒

Confirm Password
🔒

Update Password

Go Back

When the user wants to change details of his/her profile, he/she can update the profile details via this page. It is accessed via the “My Profile” button in the profile window of the home page. The user can change the username or password.

Username has been updated!

My Profile

Current Username:

Change Username

Update Username

Current Password:

Change Password

Confirm Password

Update Password

Go Back

As long as the username is not already taken, the user is free to change his/her name.

That username already exists

My Profile

Current Username:

Change Username

Update Username

Current Password:

Change Password

Confirm Password

Update Password

Go Back

If the user attempts to update his/her username with a name that is already taken, the message “That username already exists” displays.

Passwords do not match

My Profile

Current Username:

Change Username



Update Username

Current Password:

Change Password



Confirm Password



Update Password

Go Back

If the user attempts to update his/her password with two different passwords in “change password” and “confirm password” fields, the message “Passwords do not match” displays.

Password has been updated!

My Profile

Current Username:

Change Username



Update Username

Current Password:

Change Password



Confirm Password



Update Password

Go Back

When the user enters a new password and confirms it successfully, the message “Password has been updated!” displays.

Select an place in the map where you want to place your community.
What do you want to call your community?

Enter a description about your community.

Create Community

Go Back

This page is for creation of a new community. It is accessed via the “Create New Community” button in the profile window. The user enters what he/she wants to call the community and the description of it. Eventually, the user will be able to click on the map in order to place the location of the community.

New Community Created!

C'mon Baby
DADDY NEEDS A BULLSEYE!

SQL to HTML Breakthrough
It is finally done, but there is still more

Only a test
what happens now

Hello whocares
My Profile

Create New Community

My Communities

HardcoreProgrammers
A community consisting of hardcore programmers
[Visit Community](#)

Casual Bastards
COME AT ME BROSKIS
[Visit Community](#)

Job Seekers
A group of people looking for jobs
[Visit Community](#)

Insanity Cyclers
A community of people who love to take insane bike rides
[Visit Community](#)

Only a test
The only purpose of this community is to test the rendering of an addition
[Visit Community](#)

Cult of Diddy Drop
A community of rappers that enjoy the DK Cartoon song Diddy drop Rap
[Visit Community](#)

Scroll Researchers
A community of programmers figuring out how to turn this section into a scrolling window
[Visit Community](#)

Track Stars
A community of Hardcore Runners
[Visit Community](#)

Once the community is created, it will show within the communities window on the right. Eventually, it will also show up on the map. The message “New Community Created!” is displayed. The new community is marked with a green star.

Community Hub

Posts of the community will go here

Make a new post

Description of the Community goes here

Events of the Community go here

Create a New Event

When the user clicks on “Visit Community” for a particular community, he/she will be taken to the community hub page. The hub will show the posts, events, and description of the community that the user visited.

New Post

What do you want to title your post?

What do you want to talk about in this post?

Create Post
Cancel

This page is for creating new posts. It is accessed via the “Make a new post” button on the Community hub located in the post window.

Post Created!

C'mon Baby
 DADDY NEEDS A BULLSEYE!

SQL to HTML Breakthrough
 It is finally done, but there is still more

Only a test
 what happens now

Addition of Sora
 The dream is true

Hello whocares

[My Profile](#)

[Create New Community](#)

My Communities

HardcoreProgrammers
 A community consisting of hardcore programmers
[Visit Community](#)

Casual Bastards
 COME AT ME BROSKIS
[Visit Community](#)

Job Seekers
 A group of people looking for jobs
[Visit Community](#)

Insanity Cyclers
 A community of people who love to take insane bike rides
[Visit Community](#)

Only a test
 The only purpose of this community is to test the rendering of an addition
[Visit Community](#)

Cult of Diddy Drop
 A community of rappers that enjoy the DK Cartoon song
 Diddy drop Rap
[Visit Community](#)

Scrol Researchers
 A community of programmers figuring out how to turn this section into a scrolling window
[Visit Community](#)

Track Stars
 A community of Hardcore Runners
[Visit Community](#)

After the new post is created, it is displayed in the post window. The new post is marked with a green star. Eventually, it will also show up in the corresponding community hub.

Maintenance of Standards

Concerning our ISO standard, we have arranged our data such that anything sensitive in our database is encrypted. This encryption is performed via hashing.

```
func HashPassword(password string) (string, error) {
    bytes, err := bcrypt.GenerateFromPassword([]byte(password), 14)
    return string(bytes), err
}

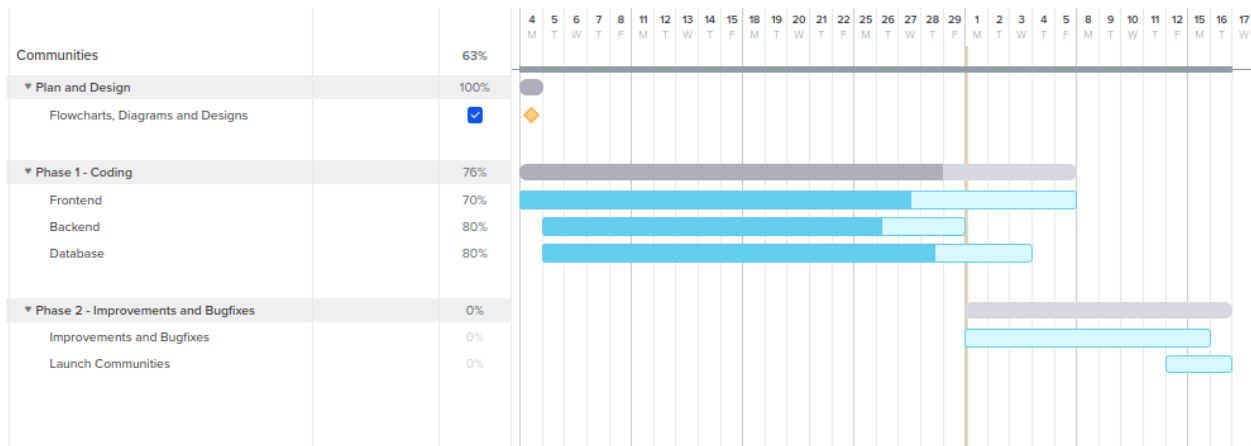
func CheckPasswordHash(password, hash string) bool {
    err := bcrypt.CompareHashAndPassword([]byte(hash), []byte(password))
    return err == nil
}
```

The functions above are responsible for handling the hashing of passwords.

For GDPR standards, all data used for the internal functionality of the application only stays with the application. Under no circumstances will we ever sell data to third parties, even when we implement ads.

Lastly, we are following the ANSI with our general approach to the project, Our application is designed to improve public health by reinforcing in person interaction between others. We do this by limiting interactions between communities to the local scope.

Updated Gantt Chart



We're currently in Phase Two of our development process. The core functionality of the application is all present, and the team is currently now dealing with mostly improvements and bugfixes. The tasks include refining the front end to look and perform better and fixing backend issues such as handling multiple users at the same time. In addition, we are currently looking to implement a few additional features that are outside of our core functionality.

References

Carlile, Liz. *Development Online: Making the Most of Social Media*. International Institute for Environment and Development, 2011, www.jstor.org/stable/resrep01460. Accessed 13 Sept. 2021.

Nair, Madhu. "Social Media Effects on Communication." *University of the People*, 7 Jan. 2021, <https://www.uopeople.edu/blog/how-social-media-affected-communication/>.

Pantalone, Maria. "Social Media and Its Impact on Communication Skills." *Infinite Growth*, 5 Nov. 2021, <https://infinitegrowth.com.au/social-media-and-its-impact-on-communication-skills/>.